Re: On-the-fly Call Graph Construction

Ben Holland <benjholla@gmail.com>

Tue 3/1/2016 1:33 PM

To: Le, Wei [COM S] <weile@iastate.edu>
Cc: Payas Awadhutkar; Zahra Hosseini; Johnston, David W [COM S]

1 attachment (1 KB)
OnTheFlyCallGraphConstruction.java;

Attached (OnTheFlyCallGraphConstruction.java) is a small example that shows the strength of on-the-fly call graph construction.

Let's say our client analysis goal is to predict the output of the program. With a CHA analysis you'd conservatively assume that any "print" method (A, B, or C's implementation of print) could be called on line 12 where we do "b2.print(c2)". The resulting call graph is attached.

With VTA + On-The-Fly Call Graph Construction we start off by not trusting any dynamic dispatches. During some iteration of the points-to analysis we find that b2 points to b1 which points to new B(). We can now add an interprocedural call edge in our dynamic transitive closure graph to B.print. Which means the flow from c2 to the "object" parameter in B.print is now connected, so once c2 comes up in the worklist we can propagate information from c2 to object and eventually to System.out.println. The resulting call graph (after a fixed point has been reached) is also attached.

I'd say that both approaches are sound (assuming your points-to analysis is sound), but the VTA is more precise than CHA. Meaning VTA produces a subset of CHA.